# MobilisGroups: Location-based Group Formation in Mobile Social Networks

Robert Lübke, Daniel Schuster, Alexander Schill
*Computer Networks Group*
*Technische Universität Dresden*
*Dresden, Germany*
*robert.luebke@mailbox.tu-dresden.de, daniel.schuster@tu-dresden.de, alexander.schill@tu-dresden.de*

***Abstract*—Location restrictions in Mobile Social Networks are often used to realize the notion of places as well as for proximity-based friend or interest matching. In this work, we extend this by the ability to create and manage groups with location and time restrictions regarding their visibility and ability to be joined. Its main contribution is the design and reference implementation of the reusable MobilisGroups service as part of the Mobilis platform, an XMPP-based service environment for developers of mobile social software. Location-based group formation furthermore links the physical and the virtual world by creating incentives to be at a certain place at a certain time and complements the two approaches mentioned above.**

***Keywords*-social networks; group formation; XMPP; social software; mobile computing; Android**

## I. Introduction

Mobile Social Networks (MSN) supporting location-based interaction among persons like Foursquare [1], Gowalla [2] and Facebook Places [3] are getting more and more popular for users of high-end mobile phones like the iPhone or Android phones. Meanwhile, there is a growing interest in the research area of mobile social software leading to a number of innovative functionality not yet available in commercial MSNs like peer-to-peer friend-of-friend detection in VENETA [4] or integration of multiple sensors like in CenceMe [5].

The Mobilis project [6] tries to support developers of such mobile social software with a reusable toolkit providing functionality like direct and group communication, import of contacts from existing social networks, location sharing, proximity detection, media sharing as well as shared editing of XML objects.

In this paper we focus on another facet of the Mobilis platform, i.e., the creation, management and usage of groups. While groups are now an integral part of commercial social network services, there are only a few systems like Urbiflock [7] and Socialaware [8] exploiting location context to establish groups. We build upon these approaches to create our group management service for the Mobilis platform. While MobilisGroups is presented here as part of the bigger research effort of Mobilis to support developers of mobile social software, there is also a new aspect of MobilisGroups worth discussing in this paper: It supports temporal as well as spatial restrictions for the visibility of groups and ability to be joined. Up to our knowledge, there is currently no other system supporting this functionality.

Why should such restrictions be useful? Spatial restrictions already prove their usefulness for the notion of places as used in Foursquare [1]. They provide an incentive for users to be at a physical place to login and interact with other nearby users. Time restrictions may optionally extend this to give an incentive to be at a certain place at a certain time.

In the following, we first discuss scenarios and requirements of location-based group formation (Section II). Background and related work is discussed in Section III. We define our architecture and protocols in Section IV. This concept was implemented in a running prototype as shown in Section V.

## II. Scenarios and Requirements

We envision two main scenarios that can benefit from location-based group formation. The first scenario is an event scenario where a BarCamp with 100 or more participants takes place at a certain location in a city. The organizer of the BarCamp creates an event group that starts being visible within the city 4 weeks before the event. So users passing by the BarCamp location will see the event group on the map. Participants arriving the day before will already see an icon of the event group and use it to find the correct location of the BarCamp. Joining the group is only possible inside or around the building (spatial restriction) on the day the BarCamp takes place (time restriction). Participants use the group to look at other participant's profiles or to leave messages during the event. After the event the group remains active and enables sharing of pictures or other media as well as contacting other BarCamp participants.

A second type of scenario is to build long-living location-based communities like a community of tenants living in the same big multi-apartment building. One of the tenants creates a group for the floor he lives in which is only visible and may be joined right at the place of the house within a short range. Other tenants using the app could see the new group on the map and join it. Optionally, this could require a confirmation by the group owner. Again, people can see each others' profiles, try to make contact using messages and group chat or share media files. This scenario may be extended by in-door positioning based on WLAN signal strength.

As can be seen in the two scenarios, the process of group creation and group management (**Req 1**) is essential and thus it is our main functional requirement. It covers creating new groups and events, updating and deleting own groups and inviting other users to a group. Furthermore, the service has to suggest groups (**Req 2**) to the user based on the user's current location, but also based on the group memberships of the user's friends. Moreover groups should be able to have geographical limits and time restrictions (**Req 3**) concerning visibility and ability to be joined. A permanent chat room should exist for every group (**Req 4**), so members can easily communicate with each other. As the main goal of the application is supporting the user to socialize, it must be possible to see other group members' profiles and add them as friends (**Req 5**).

Groups in our scenarios are rather static as they are created explicitly and tied to a specific location. Such static groups with fixed location can be complemented by dynamic groups created based on user interests and/or proximity of users such as in Urbiflock [7]. Thus we mention dynamic group management as additional requirement (**Req 6**) which is not yet realized by MobilisGroups. This could be added later using one of the existing approaches described in the next section.

## III. Background and Related Work

MobilisGroups is part of the Mobilis platform [6], [9], a framework built on top of the eXtensible Messaging and Presence Protocol XMPP [10] to ease the development of pervasive social computing applications. Up to now, we are supporting services for presence, session establishment and direct real-time communication, proximity detection, media sharing and shared editing of XML objects. A running XMPP server is needed to provide the basic presence and communication functionality while our own Mobilis Services run as XMPP clients connected to the XMPP server. Our client applications are currently all implemented on the Android platform while XMPP as an open XML protocol makes it easy to also support other mobile platforms in the future.

MobilisGroups is meant to be a component to establish location-based groups as part of Mobile Social Networking (MSN) applications (also called Pervasive Social Networking (PSN) [11]). We thus further investigate on the role of location, places and groups in recent MSN research.

The notion of location plays an important role in many MSN applications. Often there is some proximity-based matching of people to people like in PeopleTones [12], MobiClique [13] or VENETA [4], which can be done either at a server or using P2P technologies like short-range communication (e.g., Bluetooth). Beyond proximity, there can also be matching of location-dependent requests and offers like in PeopleNet [14], a matching of activities of nearby users like in [15] or even collection and distribution of complex user status (e.g., "dancing at a party with friends") like in CenceMe [5]. P3 Systems [16] go one step further and describe a framework for linking people-to-people-to-geographical places. While this is already close to our intention, it does not yet support our notion of groups with a defined location and location-dependent properties.

The notion of places can be found in a number of popular commercial MSNs like Foursqare [1], Gowalla [2] or Facebook Places [3]. Visitors of places are able to leave messages, to earn points or even become the virtual mayor of a place. Thus places already include a special type of group, i.e., the list of persons that already visited the place in some time period (e.g., within the last 6 months).

Research work using the notion of groups can be found in the Cluestr [17] system which quickly clusters contacts from a personal social network to form groups like coworkers, family and friends. Graph clustering algorithms are used to find the clusters based on the finding that groups are highly interlinked. SocialAware [8] establishes groups based on proximity. One of the demo applications is SocialAware-Tunes where users meet in front of a jukebox. The jukebox recognizes the users' Facebook IDs using Bluetooth connectivity, loads the profiles and tries to establish a playlist that matches all of the group participants' preferences. Urbiflock [7] is a framework to manage dynamic user groups based on user profiles and physical proximity. The user can create groups such as "nearbyBadmintonGroup" consisting of all users or friends interested in playing badminton that are currently in proximity to the user. Such functionality is complementary to the type of location-based groups we are targeting with MobilisGroups.

Both approaches - places as well as dynamic groups - can also be used as a starting point to create a static group tied to a specific location. We will demonstrate this for places later on by using the Foursquare API while dynamic groups are not yet supported but could be added accordingly. In the following, we will have a closer look at the design and implementation of MobilisGroups.

## IV. Design of MobilisGroups

### A. Architecture

To realize MobilisGroups, we integrated it in the centralized architecture of the Mobilis platform [6], [9] with a dedicated Grouping Service as can be seen in Figure 1. Regarding the question of how centralized a MSN should be, Mokhtar et al. [15] define three degrees of MSN middleware deployment, i.e., fully centralized, semi-distributed and fully distributed. They show that the centralized solution is the best choice for MSNs in terms of user satisfaction [11] while this approach clearly has the disadvantage of the need to build a fixed, reliable infrastructure first. Nevertheless, it would be interesting to see how our Mobilis services

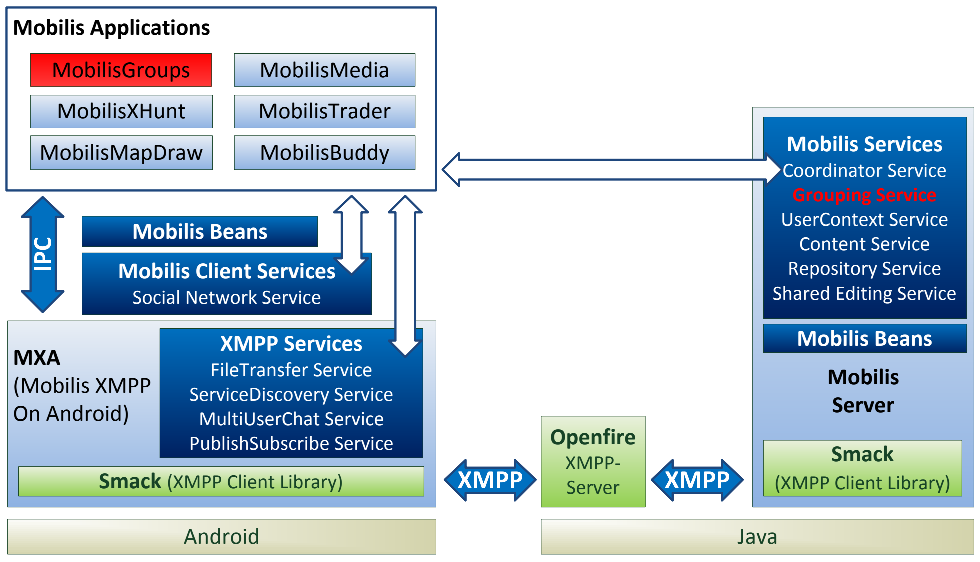


Figure 1. Integration of MobilisGroups components (red) into the existing Mobilis architecture.

could be realized in a semi-distributed or fully distributed deployment as this implies additional mechanisms to deal with incomplete information and inconsistency.

As already mentioned, we use XMPP for communication between clients and server. XMPP provides feature-rich communication and service management protocols and functions and has therefore been selected as the basis of the service platform.

Android has been selected as the best fitting mobile device platform for software development in the project. For the actual communication we chose the Smack XMPP client library [18] and adapted it to be used on the Android system. Like all other Mobilis applications, MobilisGroups uses inter process communication to access the MXA application (Mobilis XMPP on Android), that manages the XMPP communication.

Regarding the server side, the services running on the Mobilis Server act as XMPP clients themselves, thus an off-the-shelf XMPP server like OpenFire can be used as mediator. This requires clients and the Mobilis services to have their own XMPP ID (e.g., alice@xmpp.org/NexusOne, mobilis@xmpp.org/Grouping) to be able to exchange messages via the XMPP server.

## B. Service Discovery

To set up communication between a mobile client and the Grouping Service, the XMPP ID of the Grouping Service first has to be determined. We use the XMPP Service Discovery extension (XEP-0030 [19]) for this purpose. As already described in [6], the Coordinator Service, which has a well-known XMPP ID, works like a service directory and returns the ID of the Grouping Service. This is done using so-called info/query (IQ) messages. IQ messages are an RPC-like mechanism within XMPP allowing for any XML request/response pairs to be transported directly to an XMPP client using his ID.

Any Android app willing to use location-based group formation runs on the mobile client and invokes the MobilisGroups service. It communicates with the Mobilis Server by sending XMPP messages with the help of MXA library. On the server side, the Grouping Service receives these messages that are forwarded by the XMPP server. The Grouping Service manages all groups and memberships using a database. Additional services can be used for sharing location (User Context Service) and media (Repository Service and Content Service) to realize the functionality in our application scenarios. These services are described in [6].

## C. Communication Protocol

For the actual communication protocol we do not use existing standardized XMPP extensions but define our own extension with the help of custom IQ (info/query) messages. In the following, one example of these custom IQs is discussed.

For creating a new group the client uses the *GroupCreate* IQ. It contains all important data of the group, e.g., the group name, a description, a homepage, an address, the geographical position, defined restrictions and privacy information. The structure of an example *GroupCreate* IQ is shown in Figure 2 on the left side.

If a client wants to create a new group, a *GroupCreate* IQ with the type *set* has to be sent to the Grouping Service where the group is created and saved in a database. As a

```xml
<iq id="create001" type="set" from="client@xmpp/MXA"
        to="mobilis@xmpp/Grouping">
    <group-create
            xmlns=" ... #services/GroupingService">
        <name>Faculty Computer Science Dresden</name>
        <description>Students and other members of
        the Faculty Computer Science, University of
        Technology Dresden</description>
        <address>Nöthnitzer Straße 46, 01187
        Dresden</address>
        <longitude_e6>13723083</longitude_e6>
        <latitude_e6>51025733</latitude_e6>
        <restriction type="join"
            radius="1000"
            latitude_e6="51025733"
            longitude_e6="13723083"
            starttime="1284508800000"
            endtime="1284595200000"/>
        ...
        <privacy>1</privacy>
        <link>http://www.inf.tu-dresden.de</link>
    </group-create>
</iq>
```

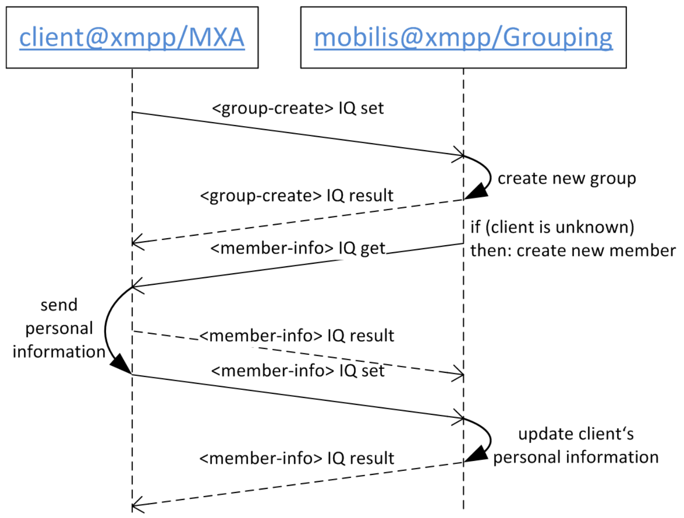


Figure 2. *GroupCreate* IQ and communication flow.

response the client receives an IQ with the type *result* which contains the URI of the newly created group. If the client is unknown to the Grouping Service, then a new member is created and further information about the client is requested with another custom IQ, the so-called *GroupMemberInfo* IQ.

The full communication flow between client and server in the group creation process is shown in Figure 2 on the right side.

### D. *Integration of external social networks*

As already mentioned in Section III, places provided by Gowalla, Foursquare or Facebook Places are a good starting point for location-based groups. All these social networks provide their data to external applications via application programming interfaces. Among others, there is a request for venues near a given geographical position, which can be used by MobilisGroups. These venues or places can then be used as a starting point to create a location-based group or event.

To meet the requirement of integrating external social networks, we included a Foursquare overlay, which the user can select to activate it. We chose Foursquare for various reasons: At the moment it has more users than Gowalla and thus more recorded venues and a better geographical precision. Furthermore, it does not require an authorization before using the API, which makes it easier to use.

The integration of external social networks is implemented in an overlay architecture. Every social network has its own layer, which can overlay the normal map view if the user activates it. This approach leads to a better extensibility. A correspondent layer for other networks can be added easily.

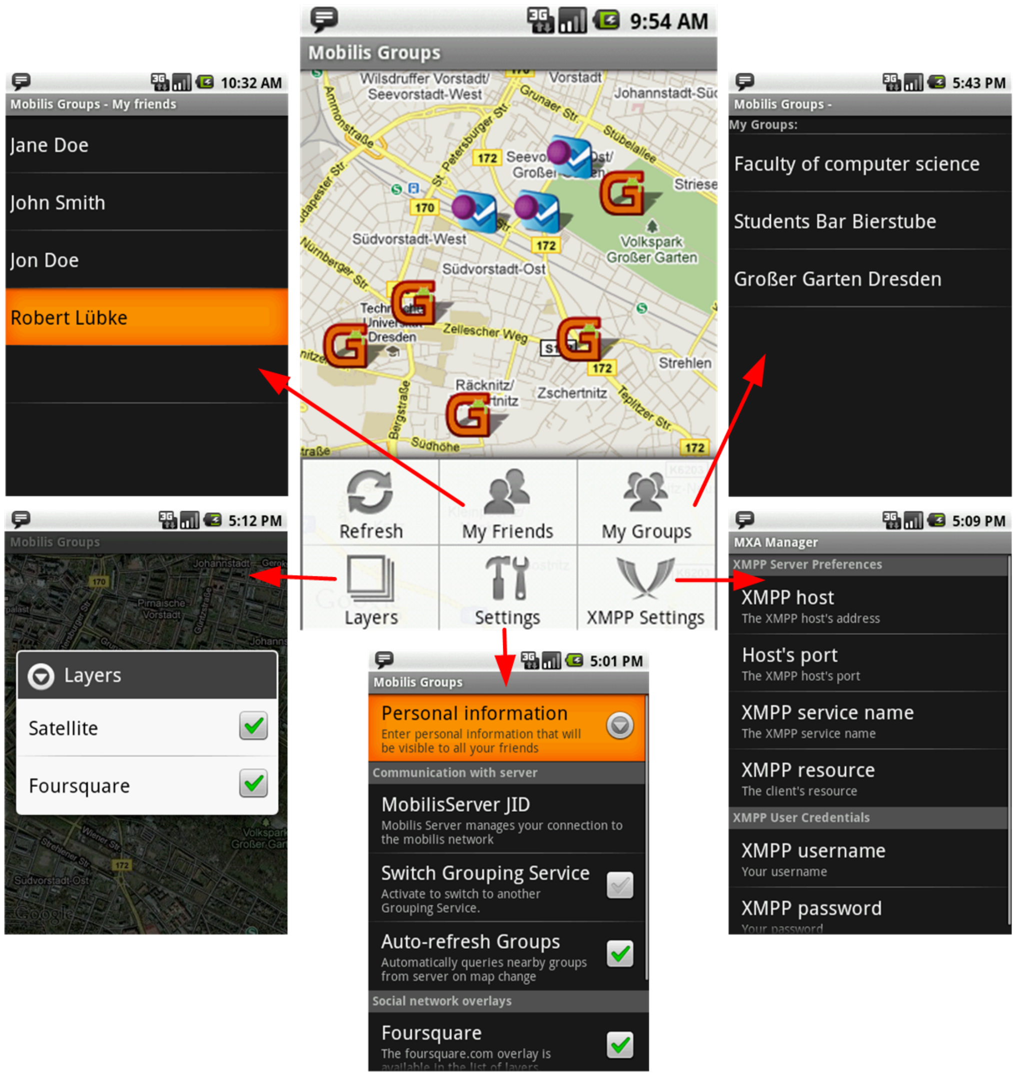
The social network integration is strictly realized on the client side, because sending requests to the Foursquare API and parsing the lightweight JSON formatted response is not too complex for a mobile client. Another possible approach is establishing a corresponding social network service in the MobilisServer, that manages the usage of multiple social network APIs for the mobile client.

## V. Implementation

To prove the feasibility of our approach, we implemented a running prototype using Android and Java on the client side as well as Java on the server side. Figure 3 shows some of the Android Activities from the graphical user interface of the prototype. In the center one can see the main view of the application: The map view. Groups and items from external social networks are visualized on the map. From this view the user can start all other main functions by clicking the correspondent menu entries.

If the application is started for the first time, the *XMPP Server Preferences* (bottom right) and application settings (bottom) have to be checked. The map view can be customized by choosing which layers should be displayed (bottom left). Clicking the entry *My Friends* opens the XMPP contact list (top left). Additional personal information is shown by selecting an XMPP buddy. This also includes the group memberships of the correspondent contact. Choosing *My Groups* in the map view's menu opens a list of all own group memberships (top right). All administrative functions are accesible here. It is possible to show details, leave, update and delete a group, open the group's multi-user chat room and invite a friend to a group. Besides the special group attributes the *Group Information* view lists all registered members. By choosing a member, further information is presented to the user again. This approach enables browsing the group and member data.

To create a new group the user has to tap the exact position on the map and a new dialog for group creation shows up. All neccessary and additional data of the group has to be



Jane Doe

John Smith

Jon Doe

Robert Lübke

Faculty of computer science

Students Bar Bierstube

Großer Garten Dresden

Mobilis Groups

Personal information
Enter personal information that will be visible to all your friends

Communication with server

MobilisServer JID
Mobilis Server manages your connection to the mobilis network

Switch Grouping Service
Activate to switch to another Grouping Service.

Auto-refresh Groups
Automatically queries nearby groups from server on map change

Social network overlays

Foursquare
The foursquare.com overlay is available in the list of layers

XMPP Server Preferences

XMPP host
The XMPP host's address

Host's port
The XMPP host's port

XMPP service name
The XMPP service name

XMPP resource
The client's resource

XMPP User Credentials

XMPP username
Your username

XMPP password
Your password

Figure 3. Screenshots from the user interface of MobilisGroups.

specified here and the group can be created. Assumed that all spatial and temporal restrictions are fulfilled, a user walking by can see it on the map, look at the group profile, use the group's multi-user chat room and join it.

Thus our prototype fulfills all the requirements mentioned in Section II except dynamic groups which is left for further study. We also tested the scalability of our approach by automatically creating 1000 groups on a 5 km square area with a visibility radius between 20 m and 200 m each. With our simple client-server setup on two machines this took about 30 ms per group to be created successfully. In a second test we simulated several random walks within the predefined square sending a request for new groups every 10 m. Not surprisingly, the processing time for a group query depends linearly on the number of groups in the visible area and ranges between 51 ms (100 groups visible) to 495 ms (1000 groups visible).

## VI. Conclusion

The main idea of this paper is to provide a reusable location-based group management service for developers of Pervasive Social Computing applications. These groups should support time and location restrictions on visibility and ability to be joined to provide incentives to be at a certain place at a certain time in the physical world. We showed a client-server architecture and XMPP-based protocols for this purpose. The protocols and prototype implementation are freely available on our Sourceforge page [9] and can be reused by other applications. It is one of the building blocks of our Mobilis platform with the goal to support developers of mobile social software on the Android platform. We are constantly refining and extending the framework and want to encourage the community to use it for their own research projects.

As we only proved feasibility of the approach by implementation and small-scale evaluation, there remains work to be done to further evaluate location-based group formation with real users. The question remains open, if location and time-restricted groups will be adopted by users of mobile social networks. Another interesting direction of future research is to link the two worlds of static and dynamic groups. As dynamic group creation exploits user proximity, the location information already available in MobilisGroups can also be used to create groups of nearby users.

## ACKNOWLEDGEMENT

We want to thank the many contributors of the Mobilis platform for their conceptual and implementation work. Alexandre Skyrme, Thomas Springer and Markus Endler provided help regarding related work. Daniel Esser, Maximilian Walther and the anonymous reviewers helped to improve this article with their feedback.

## REFERENCES

[1] foursquare, Inc., “foursquare,” http://foursquare.com/, 2010.

[2] Gowalla, Inc., “Gowalla,” http://gowalla.com/, 2010.

[3] Facebook, Inc., “Facebook Places,” http://www.facebook.com/places/, 2011.

[4] Marco von Arb, Matthias Bader, Michael Kuhn, and Roger Wattenhofer, “VENETA: Serverless Friend-of-Friend Detection in Mobile Social Networking,” in *4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Avignon, France, 2008.

[5] Emiliano Miluzzo, Nicholas D. Lane, Kristof Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell, “Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application,” in *The 6th ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Raleigh, NC, USA, 2008.

[6] Daniel Schuster, Thomas Springer, and Alexander Schill, “Service-based Development of Mobile Real-time Collaboration Applications for Social Networks,” in *First International Workshop on Communication, Collaboration and Social Networking in Pervasive Computing Environments (PerCol)*, Mannheim, Germany, 2010.

[7] Andoni Lombide Carreton, Dries Harnie, Elisa Gonzalez Boix, Christophe Scholliers, Tom Van Cutsem, and Wolfgang De Meuter, “Urbiflock: An experiment in Dynamic Group Management in Pervasive Social Applications,” in *First International Workshop on Communication, Collaboration and Social Networking in Pervasive Computing Environments (PerCol)*, Mannheim, Germany, 2010.

[8] Charles M. Gartrell, “SocialAware: Context-Aware Multimedia Presentation via Mobile Social Networks,” Master Thesis, University of Colorado, Department of Computer Science, 2008.

[9] TU Dresden, “Mobilis - A Service Platform for Collaborative Social Applications,” http://mobilisplatform.sourceforge.net/, 2010.

[10] “XMPP Standards Foundation,” http://xmpp.org, 2011.

[11] S. Ben Mokhtar and L. Capra, “From pervasive to social computing: algorithms and deployments,” in *ICPS ’09: Proceedings of the 2009 international conference on Pervasive services*, London, United Kingdom, 2009.

[12] K. A. Li, T. Y. Sohn, S. Huang, and W. G. Griswold, “Peopletones: a system for the detection and notification of buddy proximity on mobile phones,” in *MobiSys ’08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, Breckenridge, CO, USA, 2008.

[13] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot, “MobiClique: middleware for mobile social networking,” in *WOSN ’09: Proceedings of the 2nd ACM workshop on Online social networks*, Barcelona, Spain, 2009.

[14] M. Motani, V. Srinivasan, and P. S. Nuggehalli, “PeopleNet: engineering a wireless virtual social network,” in *MobiCom ’05: Proceedings of the 11th annual international conference on Mobile computing and networking*, Cologne, Germany, 2005.

[15] S. B. Mokhtar, L. McNamara, and L. Capra, “A middleware service for pervasive social networking,” in *M-PAC ’09: Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, Urbana Champaign, Illinois, 2009.

[16] Quentin Jones and Sukeshini A. Grandhi, “P3 Systems: Putting the Place Back into Social Networks,” *IEEE Internet Computing*, vol. 9, pp. 38–46, 2005.

[17] R. Grob, M. Kuhn, R. Wattenhofer, and M. Wirz, “Cluestr: mobile social networking for enhanced group communication,” in *GROUP ’09: Proceedings of the ACM 2009 international conference on Supporting group work*, Sanibel Island, Florida, USA, 2009.

[18] Ignite Realtime, “Smack API,” http://www.igniterealtime.org/projects/smack/, 2011.

[19] Joe Hildebrand, Peter Millard, Ryan Eatmon, and Peter Saint Andre, “XEP-0030: Service Discovery,” XMPP Standards Foundation, Tech. Rep., 2008.